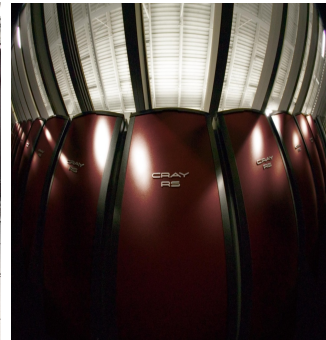


*Exceptional service in the national interest*



## XPRESS Project Update

Ron Brightwell, Technical Manager  
Scalable System Software Department



Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

# Agenda

<b>10:00 – 10:15</b>	• Project Overview	• Ron Brightwell
<b>10:15 – 10:40</b>	• Runtime System	• Thomas Sterling
<b>10:40 – 10:50</b>	• OS	• Ron Brightwell
<b>10:50 – 11:10</b>	• Performance Infrastructure	• Al Malony
<b>11:10 – 11:20</b>	• Legacy Application Support	• Barbara Chapman
<b>11:20 – 11:30</b>	• Applications	• Mike Heroux

# Project Goal

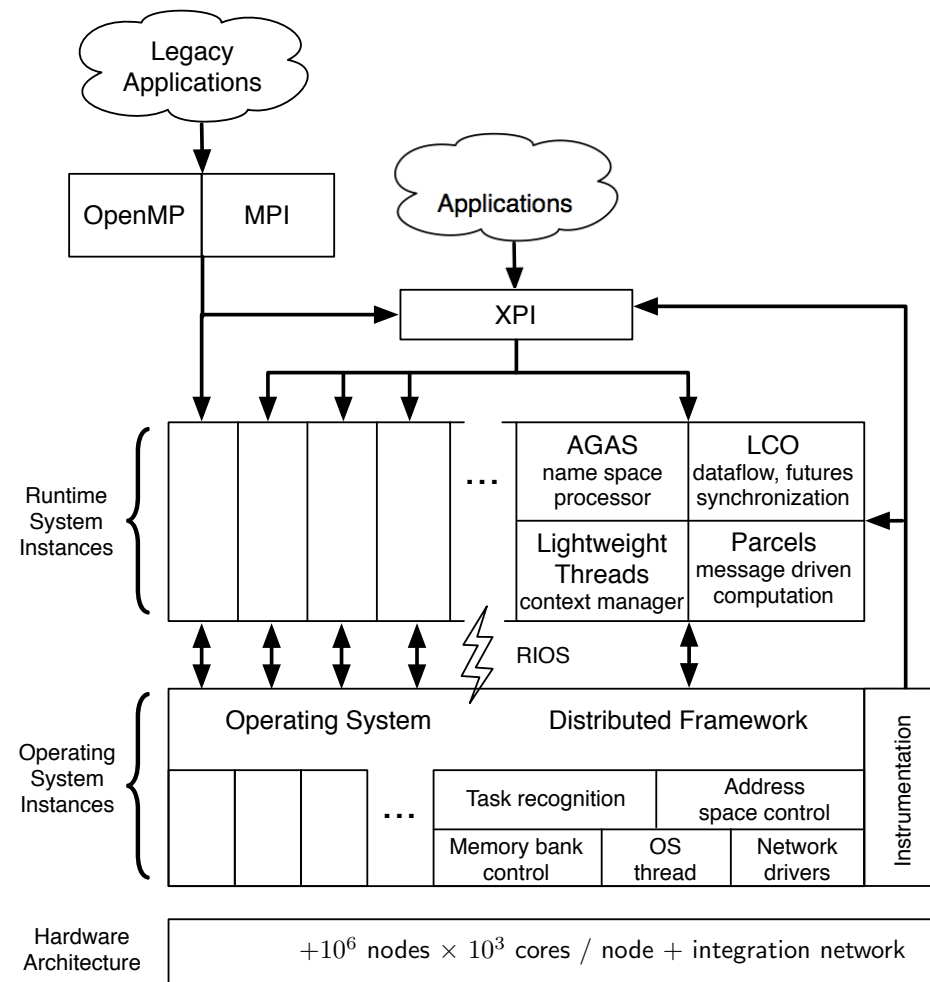
- R&D OpenX software architecture for exascale computing
- Four thrust areas
  - HPX runtime system based on the ParalleX execution model that supports dynamic resource management and task scheduling
  - L XK lightweight operating system based on the Kitten OS that exposes critical resources to HPX runtime system
  - Runtime Interface to OS (RIOS) definition and description of the interaction between HPX and L XK
  - Support for legacy MPI and OpenMP codes within OpenX



# Uniqueness of XPRESS

- Guided by ParalleX holistic parallel execution model
- Open-X software spans entire stack
  - Lightweight OS designed for dynamic adaptive runtime systems (LXK)
  - Runtime Interface to the OS (RIOS)
    - Enables efficient interaction between runtime and OS for event-driven adaptivity
  - Dynamic, adaptive runtime system (HPX)
  - Integration of performance instrumentation and control throughout software stack (APEX, RCR)
  - Low-level application programming interface (XPI)
  - Support for native applications and legacy applications
- Targets full system capability

# OpenX Software Architecture



# Year 2 Activities

Major Task	Details
<b>ParalleX Execution Model</b>	<ul style="list-style-type: none"><li>• Finish description of ParalleX document</li><li>• Incorporate locality management and introspection</li></ul>
<b>HPX-3</b>	<ul style="list-style-type: none"><li>• Continue to update, develop, and improve HPX-3</li><li>• Implement HPX-3 with XPI</li><li>• Develop the HPX-4 threading system</li><li>• Continue developing HPX process</li><li>• Adopt IU Parcel port</li><li>• Work with RENCI to integrate power management into HPX</li><li>• Work with OU to make HPX APEX aware</li><li>• Work with SNL to implement HPX on LXX</li><li>• Work with UH to provide legacy support</li></ul>
<b>HPX-4</b>	<ul style="list-style-type: none"><li>• Implement Parcels</li><li>• Implement Local Control Objects</li><li>• Implement GAS</li><li>• Implement interface to LSU's thread package</li><li>• Implement interface via RIOS to LXX</li></ul>
<b>LXX</b>	<ul style="list-style-type: none"><li>• Deploy LXX virtual cluster environment</li><li>• Demonstrate HPX-4 on LXX</li><li>• Integrate instrumentation capability into LXX</li></ul>
<b>RIOS</b>	<ul style="list-style-type: none"><li>• Refine RIOS specification</li><li>• Protocol specification</li><li>• Use specification to guide design of interface LXX and HPX-4</li></ul>

# Year 2 Activities (con'td)

Major Task	Details
<b>APEX</b>	<ul style="list-style-type: none"><li>• Implement more robust version of APEX with HPX-4</li><li>• Create a performance data access API for evaluating performance metrics mapped for lower-level measurements during execution to allow for performance data introspection</li><li>• Develop interfaces for XPI to specify performance requirements and create performance data views</li><li>• Define performance introspection requirements and architecture</li></ul>
<b>Introspection</b>	<ul style="list-style-type: none"><li>• Develop and integrate contention/energy models into HPX and APEX</li><li>• Improve and increase data source for models by integrating into LXX</li><li>• Finish design and start implementing multi-node data collection and contention/energy models</li></ul>
<b>Legacy Migration</b>	<ul style="list-style-type: none"><li>• Implement in OpenUH for supporting OpenMP 3.1 using HPX runtime</li><li>• Evaluate performance of HPX-OpenMP and OpenUH-OpenMP using benchmarks and applications</li><li>• Enhance performance of OpenACC compiler and OpenMP 4.0 accelerator support in OpenUH</li><li>• Data-driven computation model across cluster nodes</li><li>• Develop MPI collective communication operations based on HPX/XPI operations</li><li>• Finalize runtime support for MPI libraries in HPX</li></ul>
<b>Applications (Year 2 start)</b>	<ul style="list-style-type: none"><li>• XPI mini-apps</li><li>• Fusion energy apps</li><li>• Climate science and nuclear energy apps</li><li>• Collaboration with Co-Design Centers</li><li>• Collaboration with other X-Stack projects</li></ul>
<b>Software Integration</b>	<ul style="list-style-type: none"><li>• Develop plan for integrating software components</li><li>• Deploy and support application development and evaluation testbed</li></ul>

# Status

- Met all Year-1 milestones and deliverables
- On track to meet all Year-2 milestones and deliverables
- Several key documents created and/or updated
  - Significant progress on RIOS specification
  - XPI Draft Specification version 1.0
  - Updated draft of ParalleX Execution Model specification
- Significant progress on several software components
- Several cross-institutional meetings and interactions
- Application work underway

# Publications

- Kevin Huck, Sameer Shende, Allen Malony, Hartmut Kaiser, Allan Porterfield, Rob Fowler, and Ron Brightwell. **An Early Prototype of an Autonomic Performance Environment for Exascale** In *Proceedings of the 2013 International Workshop on Runtimes and Operating Systems for Supercomputers*.
- Thomas Sterling, Matthew Anderwson, P. Kevin Bohan, Maciej Brodowicz, Abhishek Kulkarni, and Bo Zhang. **Towards Exascale Co-Design in a Runtime System**. In *Proceedings of the Exascale Applications and Software Conference*, Stockholm, Sweden, April 2014.
- Matthew Anderson, Maciej Brodowicz, Abhishek Kulkarni, and Thomas Sterling. **Performance Modeling of Gyrokinetic Toroidal Simulations for a Many-Tasking Runtime System**. In *Proceedings of the Fourth International Workshop on Performance Modeling, Benchmarking, and Simulation of High-Performance Computer Systems*, Denver, CO, November 2013.
- Timur Gilmanov, Matthew Anderson, Maciej Brodowicz, and Thomas Sterling. **Application Characteristics of Many-Tasking Execution Models**. In *Proceedings of the 19<sup>th</sup> International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, NV, July 2013.
- Matthew Anderson, Maciej Brodowicz, Thomas Sterling, Hartmut Kaiser, and Bryce Adelstein-Lelbach. **Tabulated Equations of State with a Many-Tasking Execution Model**. In *Proceedings of the Workshop on Large-Scale Parallel Processing*, Boston, MA, May 2013.

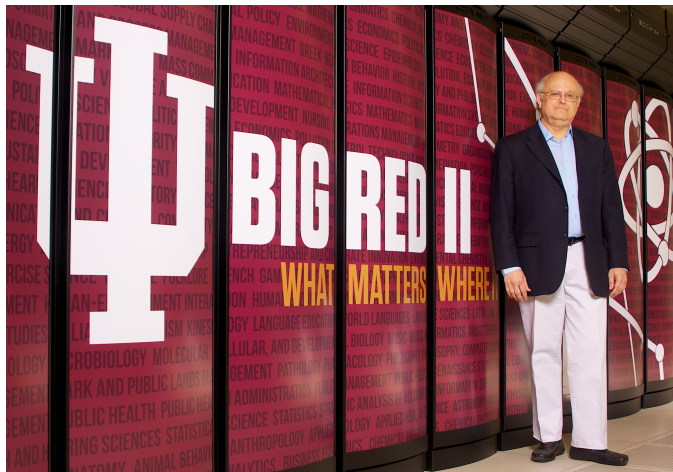
# Miscellaneous Activities

- Strong synergies with other DOE programs
  - Hobbes OS/R project
  - Execution Models
  - PSAAP-II Notre Dame project
  - DOE Design Forward
    - Ongoing discussions with AMD on HSA/XTQ
    - Sandia is a member of Heterogeneous System Architecture Foundation
- Participation on the OpenMP Architecture Review Board
  - Sandia and U. Houston are Auxiliary Members of ARB
- Completed application-facing runtime issues on xstackwiki
- Contributed to ASCR Runtime Summit

# Summary

- On schedule to meet research goals
- Working prototypes of all major components
- Dramatically improved efficiency and performance for several software components
- Demonstrated distributed message-driven computation
- Kitten/LXK on track to support HPX-4 by end of Year-2
- Preparing to do Sandia-led software integration
- Applications work underway
- Important pathfinding discoveries along the way
  - Performance limitations of model, runtime system, hardware
  - Learning how OS and runtime serve each other

# HPX: An Exascale Runtime Software Package



Thomas Sterling

Professor of Informatics and Computing, Indiana University

Chief Scientist and Executive Associate Director  
Center for Research in Extreme Scale Technologies (CREST)  
School of Informatics and Computing  
Indiana University

April 10, 2014



**CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES**

INDIANA UNIVERSITY  
Pervasive Technology Institute

# Key Strategic Points/Accomplishments

- On schedule (SOW) with research, development milestones and deliverables
- Runtime System Software Prototypes
  - Working cross-nodes parcels (message-driven) transport layer
  - GAS cross-node framework
  - Multi-threads scheduling package
- XPI specification
- XPI first function implementation with micro-apps
- Software architecture: module functionality and interfaces
- Enhanced execution model description
- Interrelationship with light-weight kernel OS
- Experimental evaluation for validation and performance
- Relevancy/engagement with other DOE Programs
  - Co-design, Execution models, OS/R, PSAAP2

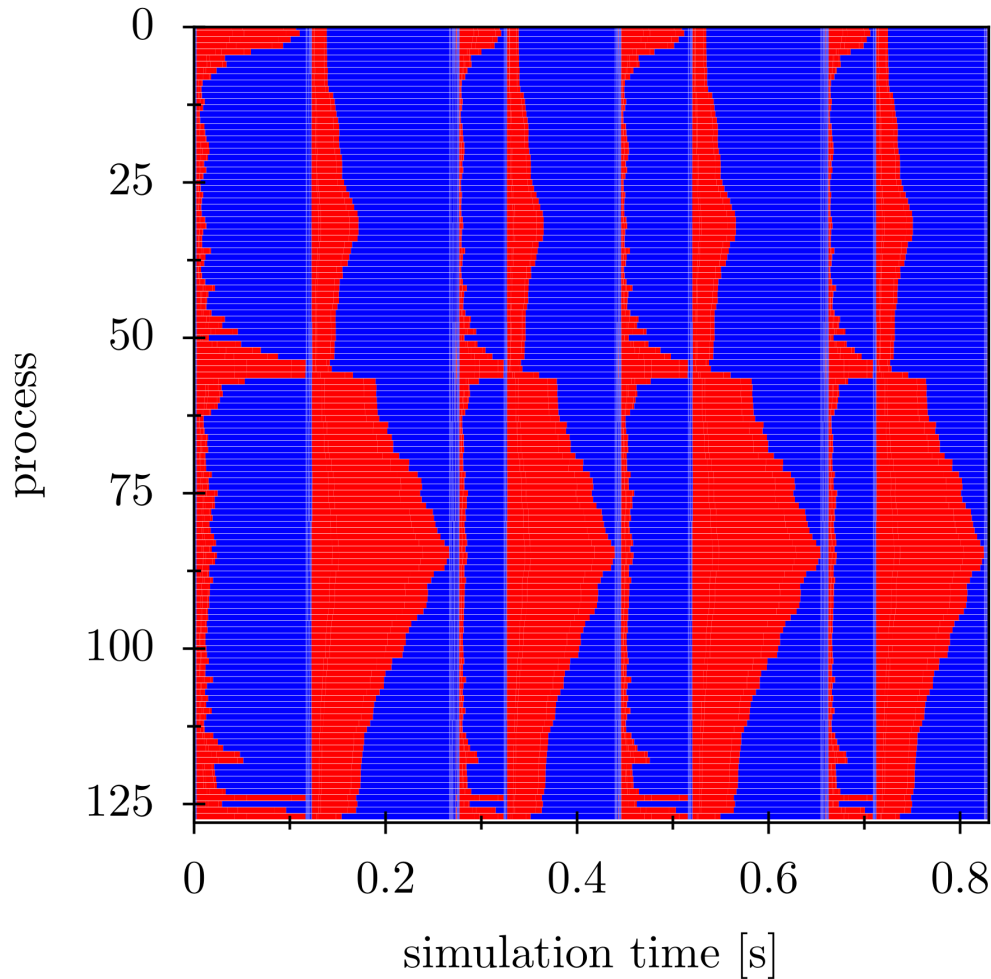


CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

---

INDIANA UNIVERSITY  
Pervasive Technology Institute

# The Negative Impact of Global Barriers in Astrophysics Codes



Computational phase diagram from the MPI based GADGET code (used for N-body and SPH simulations) using 1M particles over four time steps on 128 procs.

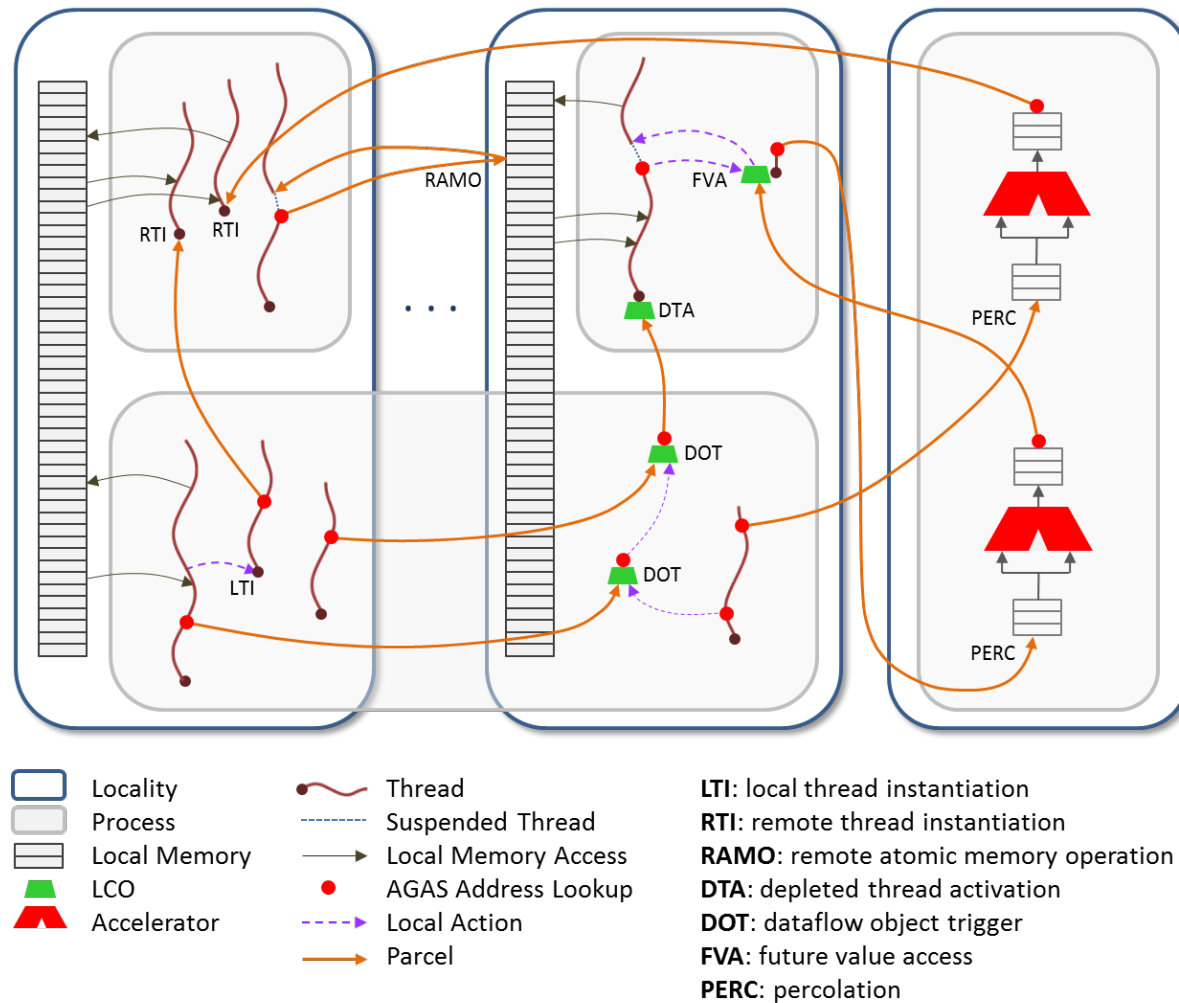
Red indicates computation  
Blue indicates waiting for communication



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

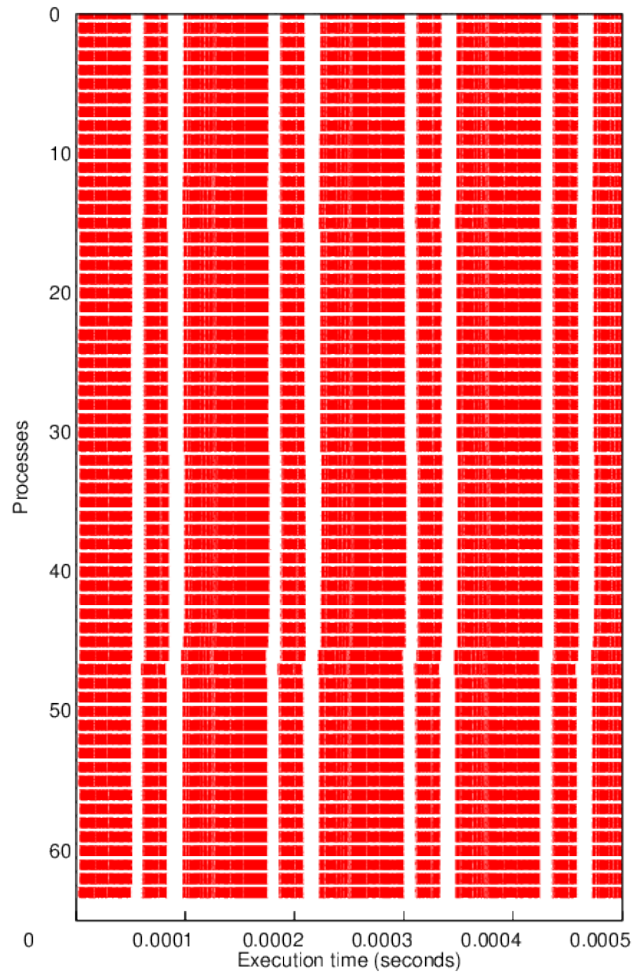
# Semantic Components of ParalleX



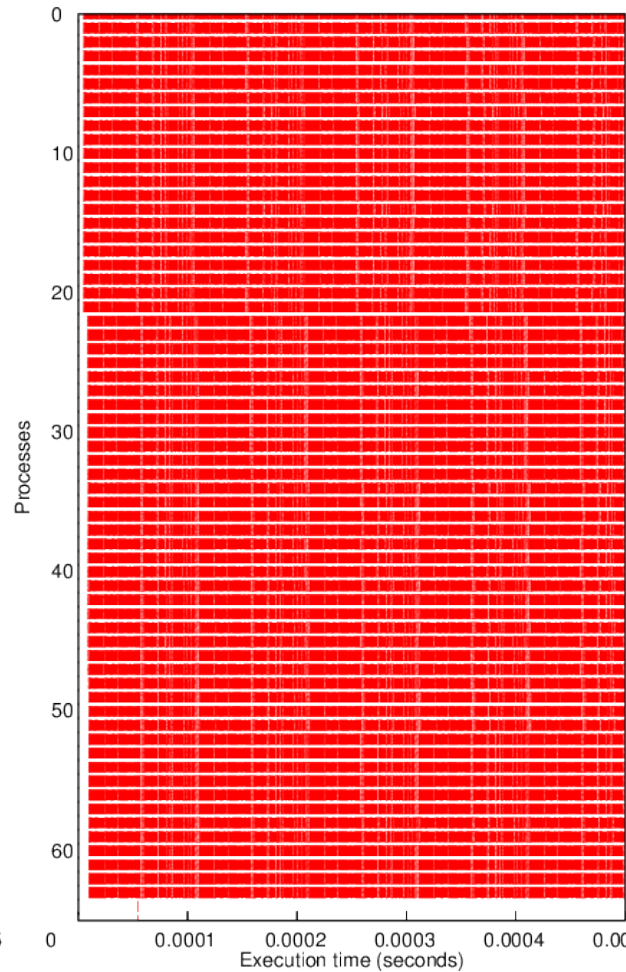
CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# Overlapping computational phases for hydrodynamics



MPI



HPX

Computational phases for LULESH (mini-app for hydrodynamics codes).

Red indicates work

White indicates waiting for communication

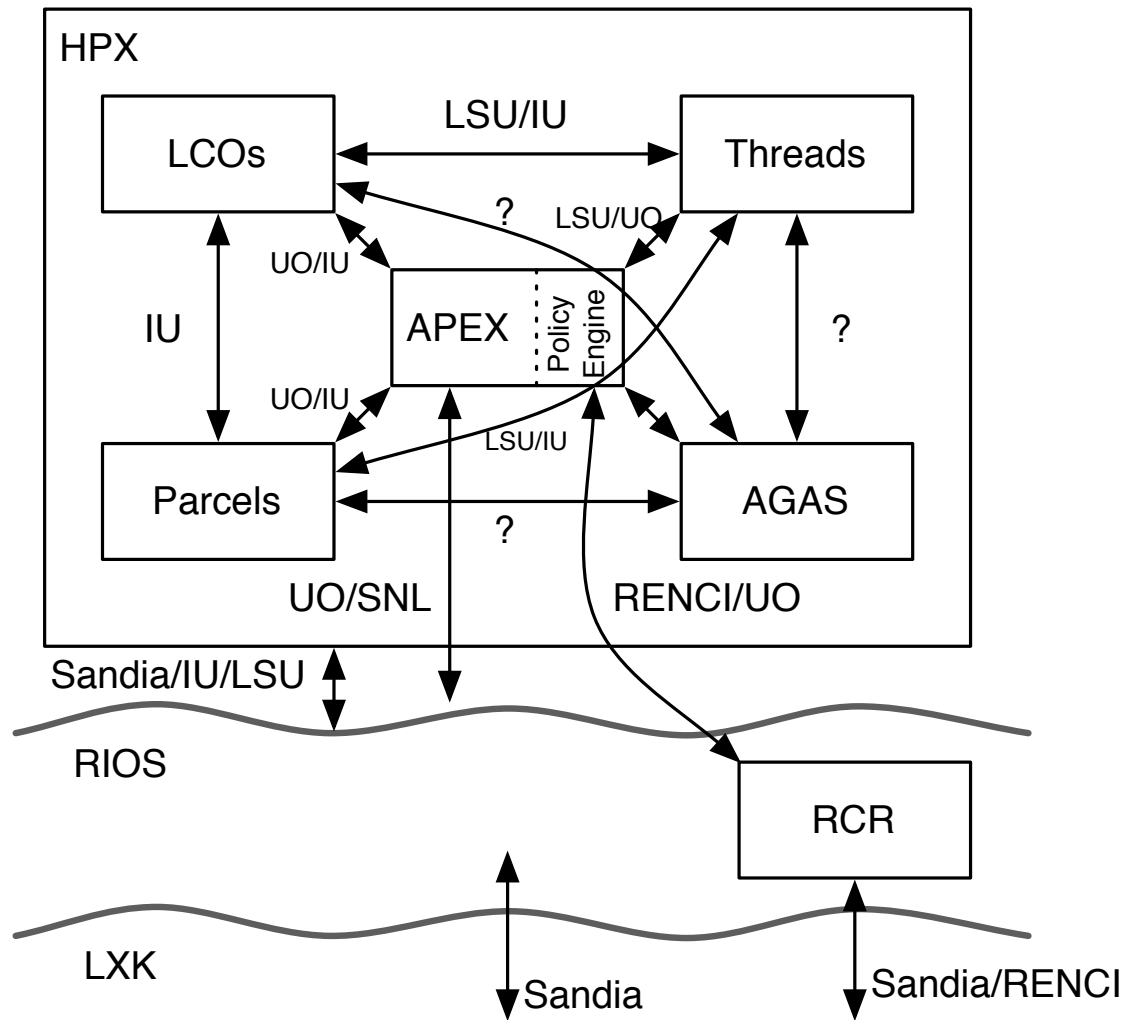
Overdecomposition: MPI used 64 process while HPX used 1E3 threads spread across 64 cores.



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# XPI



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# HPX Progress

- Overall architecture that integrates:
  - Threads
  - Local Control Objects (LCOs)
  - Performance Measurement
  - Parcels
  - Global Address Space (GAS)
- The network layer has been updated
  - Made it reentrant
  - Implemented a transport channel for large messages
- We implemented remote thread spawns & future setting over the network
- We created a new type of LCO called *versioned gates* that control groups of futures



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# HPX - Threading Subsystem

- Fully refactored code base to improve performance, modularity, and maintainability
  - Performance improved by 20-30%: overall thread (scheduling) overheads down to 900 ns per thread (create, schedule, execute, and delete thread)
  - Different schedulers are available (LIFO, FIFO, with or without thread priority, etc.)
  - Can be switched at runtime, easily extensible by user policies
- Worked with RENCI to integrate power and contention management into schedulers
- Worked with UO to instrument threading subsystem and integrate with APEX



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# HPX - Instrumentation

- HPX exposes a large set of Performance Counters
  - Uniform framework for exposing arbitrary system information
  - Examples: AGAS, threading/scheduling, parcel transport
  - Allows for runtime introspection as a precondition for runtime dynamic resource management
- APEX uses those to expose diverse set of information to tools like TAU
- HPX hooks into commercial tools like Intel® Amplifier and Intel® Inspector
  - Debugging support: memory checking, threading (deadlock) analysis, race condition checks, etc.

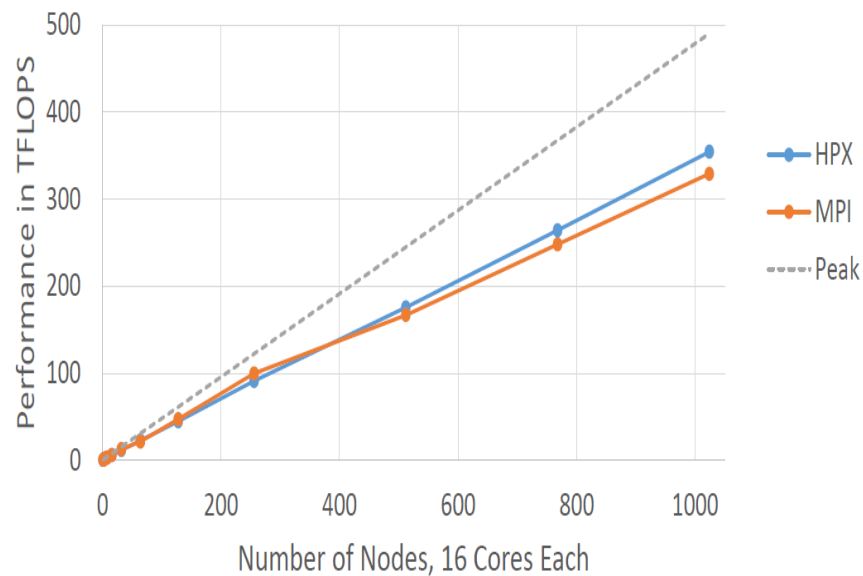


CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

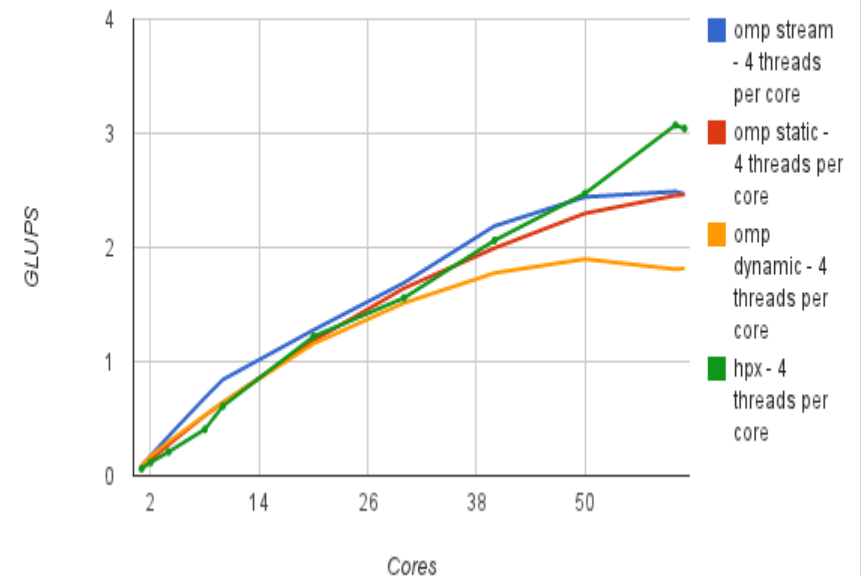
INDIANA UNIVERSITY  
Pervasive Technology Institute

# HPX – Latest Exemplar Results

Weak Scaling Results for HPX and MPI N-Body Codes  
(Host Cores only)



Jacobi Smoother - 8096x8096 - 1000 Iterations -  
Xeon Phi



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# Impact

- There is a path forward for loosely coupled integration of software components across the entire XPRESS project team
- HPX-4 runs in distributed environments now
  - Up to 512 cores (16x32) on InfiniBand
  - Up to 64 cores (4x16) on Cray Gemini
- Network performance has increased by an order of magnitude
- Applications have more data structures available to them using futures
  - That suit the needs of real-world scientific applications



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# Parcels

- Software component for cross-nodes transport layer
  - Transport layers:
    - Photon (infiniband and Gemini)
      - Experiments with unified runtime/network software architecture
    - Portals-4
      - Final transport for commonality, portability, stability
    - TCP/IP
  - Platforms
    - X86 cluster (Cutter)
    - Cray XE6/XK7 (Big Red II)
- Message-driven computation
  - Instantiates threads on remote localities (nodes)
- Global data access
  - Able to directly read and write data from all localities
- Current support for PGAS
  - Temporary static implementation
  - Will be replaced with AGAS for dynamic migration of virtual data
  - Distinction is transparent to user
- Performance studies, measurements, tradeoffs underway



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# XPI Specification (r313)

- Currently supported capabilities
  - Processes
  - Parcels
  - Threads
  - AGAS
  - LCOs
- Current and Future development
  - Locality management
  - Integrated error handling
  - Customization (thread priorities, distribution, etc)
  - Introspection (load balancing, locality management, etc)
  - IO
- Public Specification is in Beta
- Implement XPI - Released mid-March: HPXPI 0.1
  - Almost complete, fully open source implementation of XPI specification on top of HPX
  - Benefits from high performance characteristics of HPX



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# libXPI Native Implementation

- Implements (targets HPX)
  - Parcels
  - Lightweight Threading (with local task sharing)
  - PGAS (with block cyclic distribution)
  - LCOs (futures, semaphores, and gates)
  - Networking (SMP, MPI, Portals, Photon)
- In development
  - Process termination detection
  - Continuation stacks
  - Lazy thread stack binding
  - AGAS
  - Test suite
- Initial release this spring
  - Concurrent with XPI Specification 1.0 gold release



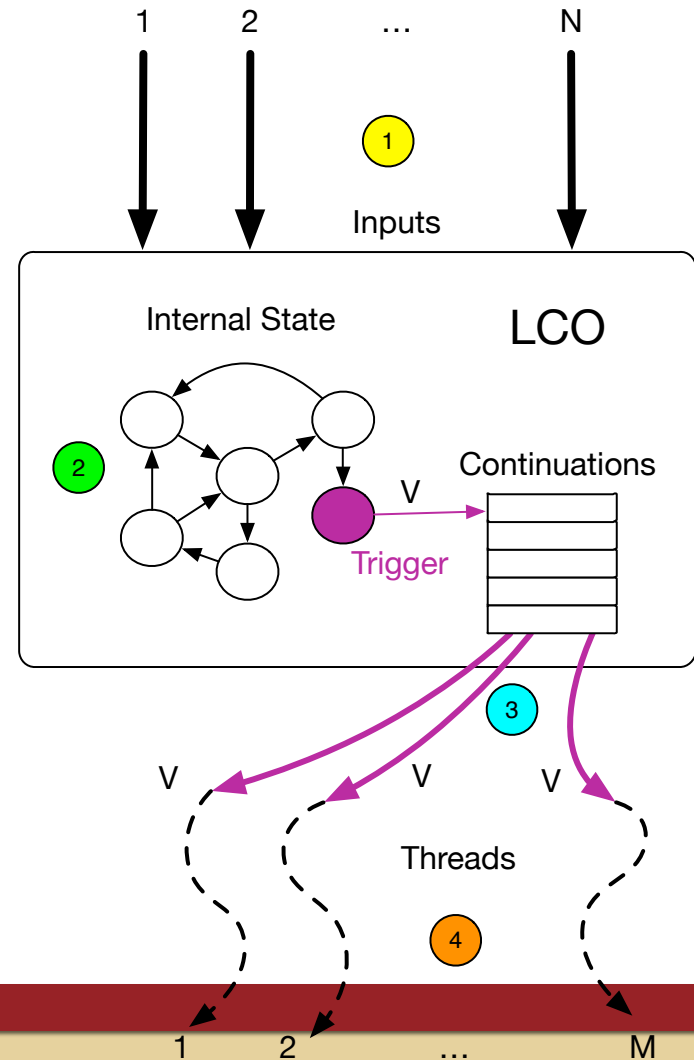
CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

# XPI LCOs

- Similar to monitors
  - Enqueue continuation parcels
  - Interface is serializable
  - User-definable polymorphic behavior

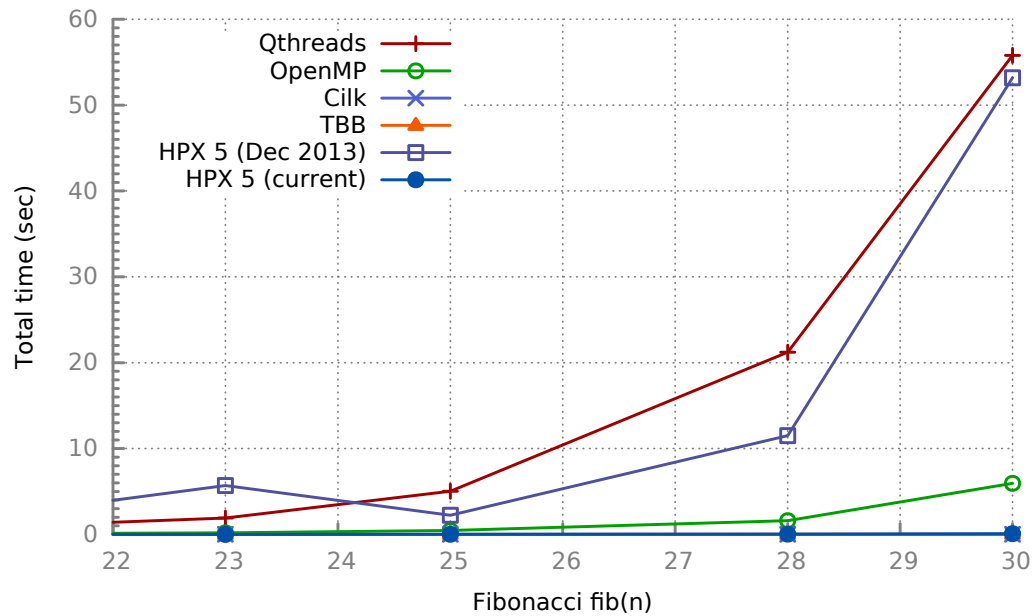
1. Inputs modify internal state
2. Trigger state copies value to continuation parcels
3. Runtime sends continuation parcels
4. Continuations instantiated as lightweight threads, or activate depleted threads.



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

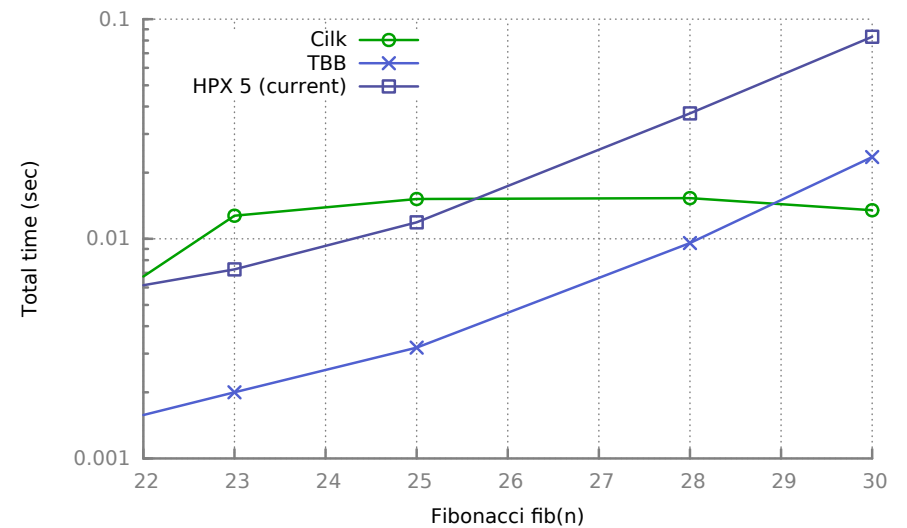
INDIANA UNIVERSITY  
Pervasive Technology Institute

# Experimental Results



All cases run on 16 cores (1 locality)

Zoom-in on best performers

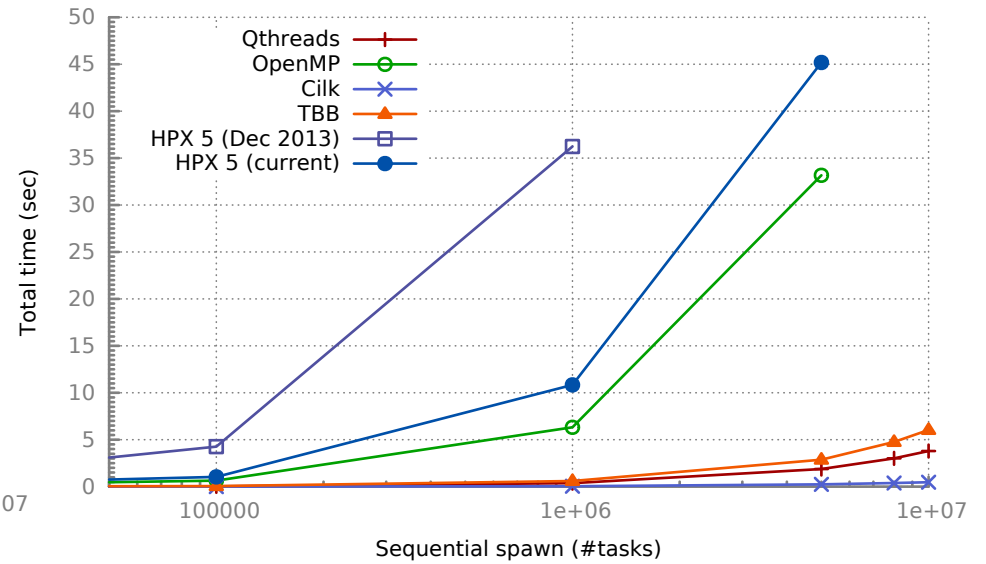
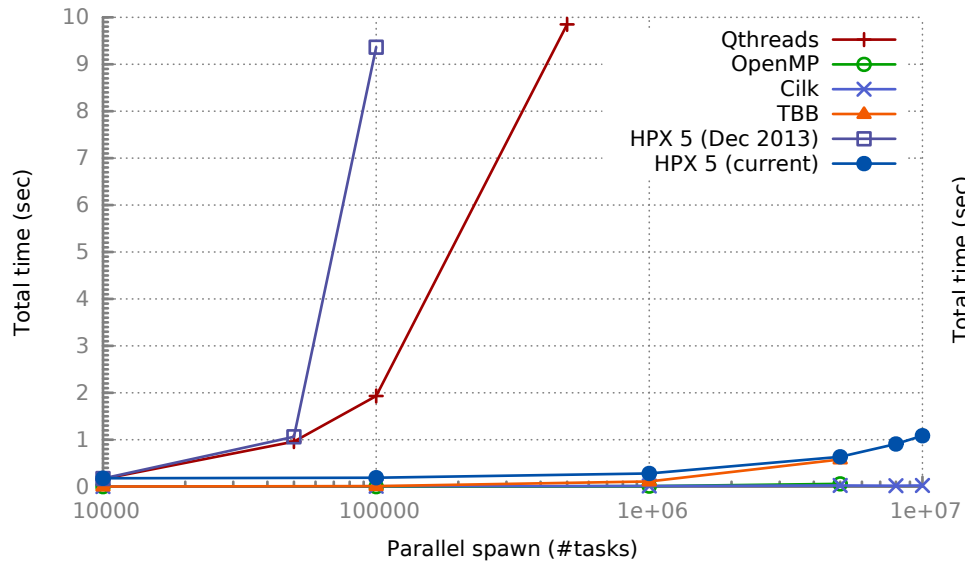


CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

*Courtesy of Matt Anderson, Indiana University*

# Experimental results on Overhead



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

*Courtesy of Matt Anderson, Indiana University*

# PRIDE of Discovery

- PRIDE – Parallel Resource Integration for Distributed Execution
- Extends ParalleX Processes up to encompass system
- Integrates L XK and HPX
- Exhibits single-system image
- Can provide global scalable extreme scale “System Global OS”
- A natural occurring consequence of sophisticated execution model and system software architecture



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

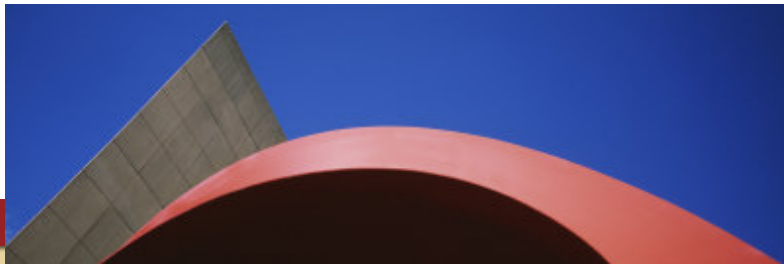
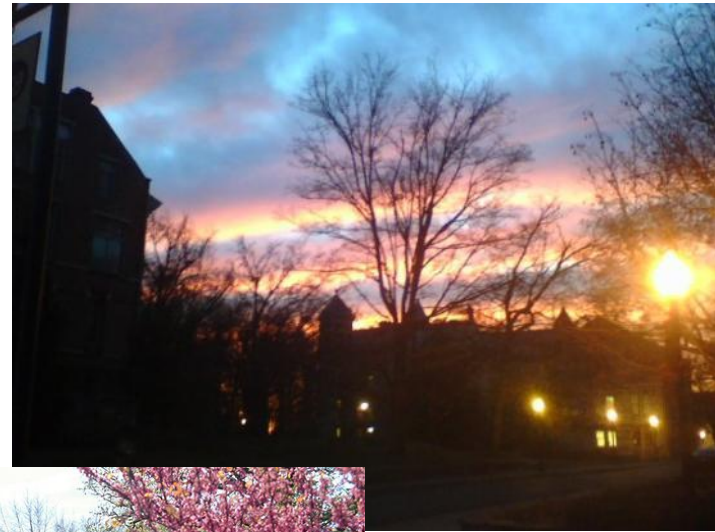
# Future Work

- Evaluating selected algorithms for refactoring solvers
  - Engagement with co-design centers
  - Mini/proxy apps
- ParalleX locality sensitivity, resource allocation, prioritization
- XPI specification refinement and implementation optimization
- HPX-4 system integration (SNL led) from LSU/IU components
- Integration of introspection components UO/UNC
  - APEX UO
  - RCR UNC
- Detailed specification of RIOS protocol
- Software Stack Build
  - LXX
  - HPX-4
  - XPI



CENTER FOR RESEARCH  
IN EXTREME SCALE  
TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute



## CENTER FOR RESEARCH IN EXTREME SCALE TECHNOLOGIES

INDIANA UNIVERSITY  
Pervasive Technology Institute

*Exceptional service in the national interest*



Photos placed in horizontal position  
with even amount of white space  
between photos and header

# XPRESS OS Update

April 10, 2014

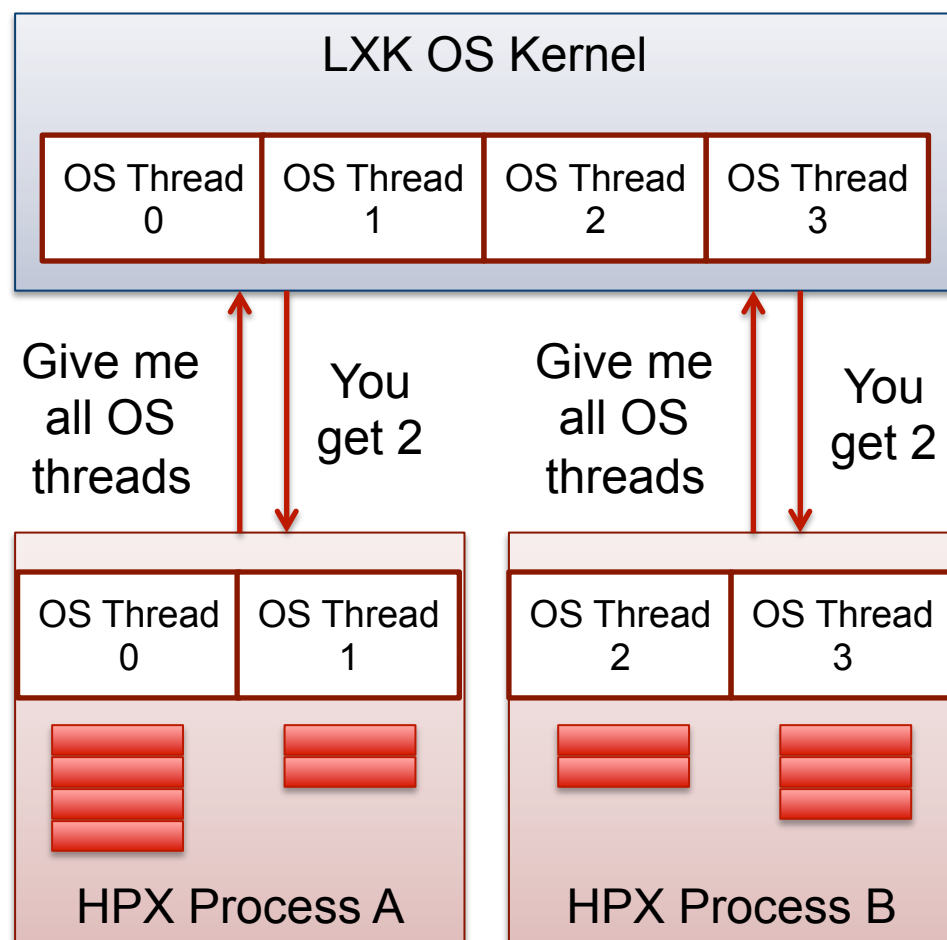


Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

# LXK/RIOS Research Goals

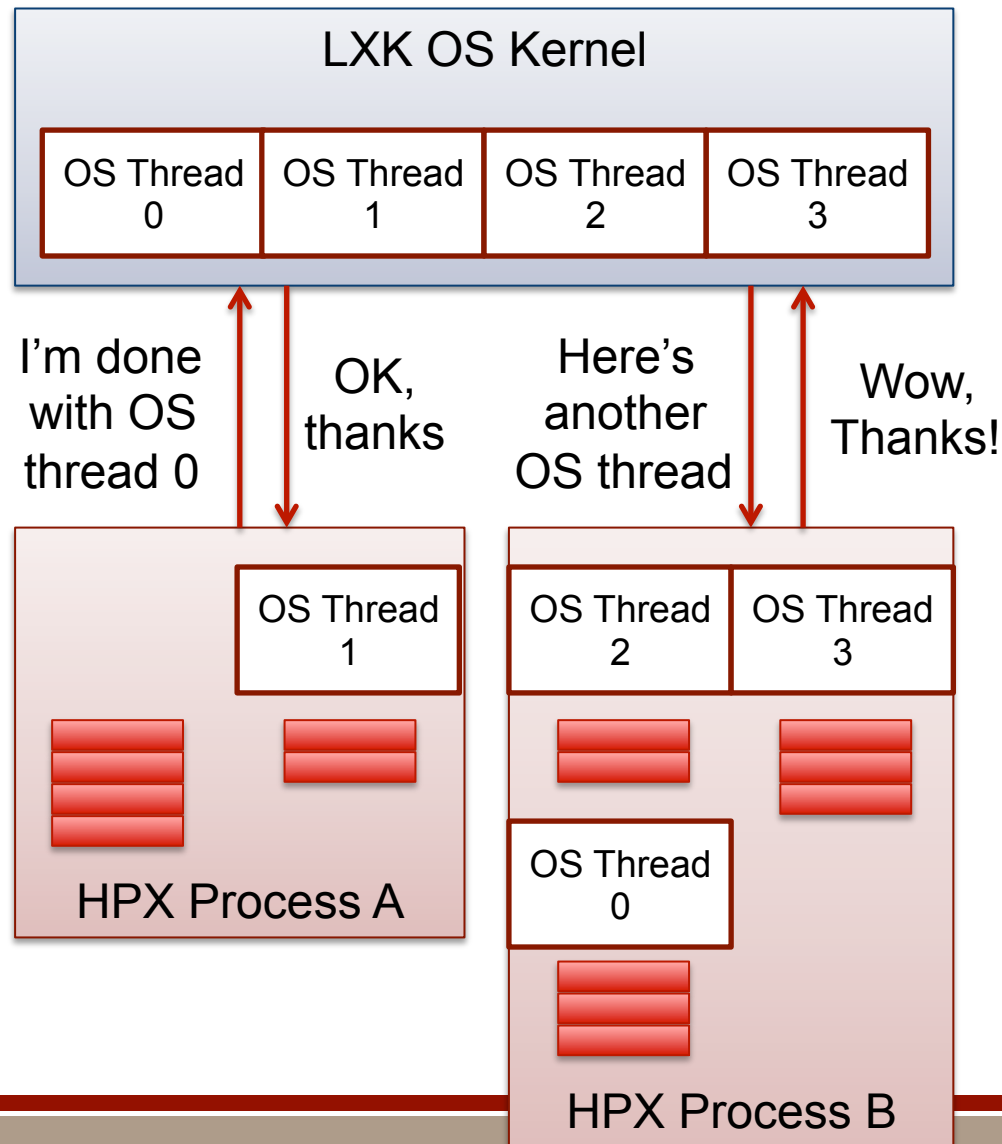
- XPRESS aims to increase synergy of compute node OS kernel and user-level runtime systems
  - Today: Runtime must work around host OS, assume worst case
  - Vision: Runtime cooperates with host OS, delegated more control
- Key RIOS drivers
  - Runtime needs guarantees about resource ownership and behavior
  - OS needs way to shift resources between multiple runtimes
  - Two-way interfaces needed for key resources
    - Runtime tells OS what it needs, OS tells runtime what it gets
    - OS remembers original request, notifies runtime if more resources become available. Notifies runtime of resources need to be reclaimed.
  - Event-based protocol to notify of dynamic events (e.g., power state change, transient error)
- LXK = Kitten + RIOS

# Two-level Thread Scheduling



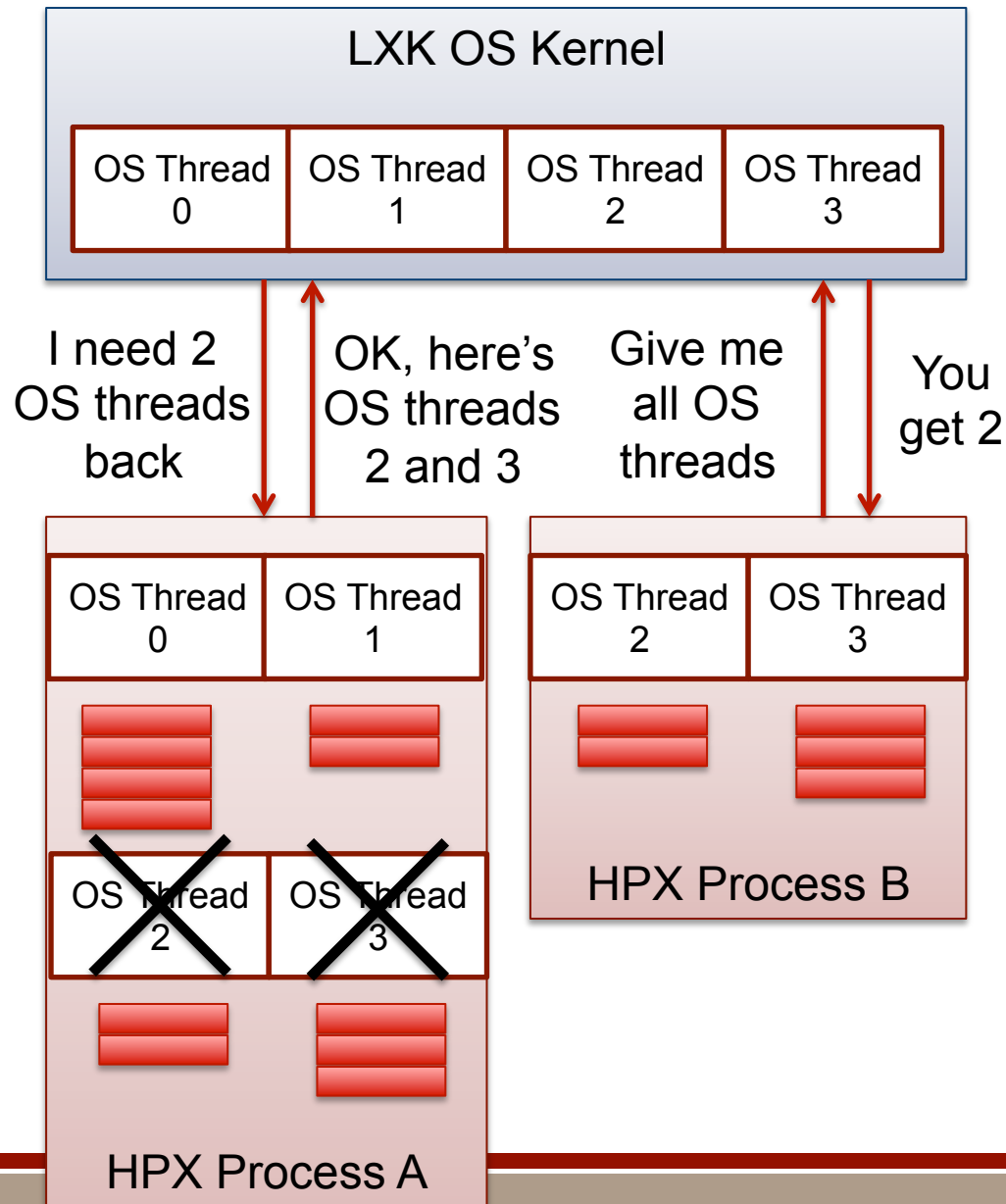
- OS threads track hardware contexts of execution (e.g., physical cores)
- Runtime requests OS threads from OS, runtime schedules its task queues onto OS threads
- RIOS defines protocol runtime uses to ask for OS threads, protocol OS uses to tell runtime what it gets

# Two-level Thread Scheduling (2)



- OS remembers original request. In this case, each HPX process could use all available OS threads but each initially allocated only two
- HPX process A gives up an OS thread, LXX decides to reallocate it to HPX process B
- HPX process B initializes a new task queue and starts scheduling tasks on the new OS thread it was allocated

# Two-level Thread Scheduling (3)



- OS can ask a runtime for OS threads back at any time
  - Runtime must cooperate or be killed by OS
  - OS gives runtime some time to react, move tasks off of the OS threads being returned
- In this example HPX Process B arrives at some time after HPX Process A; OS allocates B two OS threads that it reclaims from A

# Areas Covered by RIOS

- Legacy support services
- Job management
- Thread management
- System topology and locality
- Introspection
- Memory management
- Network interface
- File I/O
- Energy management

# RIOS Current Status

- Current draft outlines requirements and gives prose description of interfaces and protocols
- FY14 goal to formalize into a real specification that somebody could implement
- Recent focus on thread management and memory mgmt.
  - Created several working documents to drive discussions
    - “Thread Management in Kitten”, “Memory Management in Kitten”
    - IU write ups describing ParallelX process model and AGAS
  - LSU providing input on HPX-3 thread management requirements

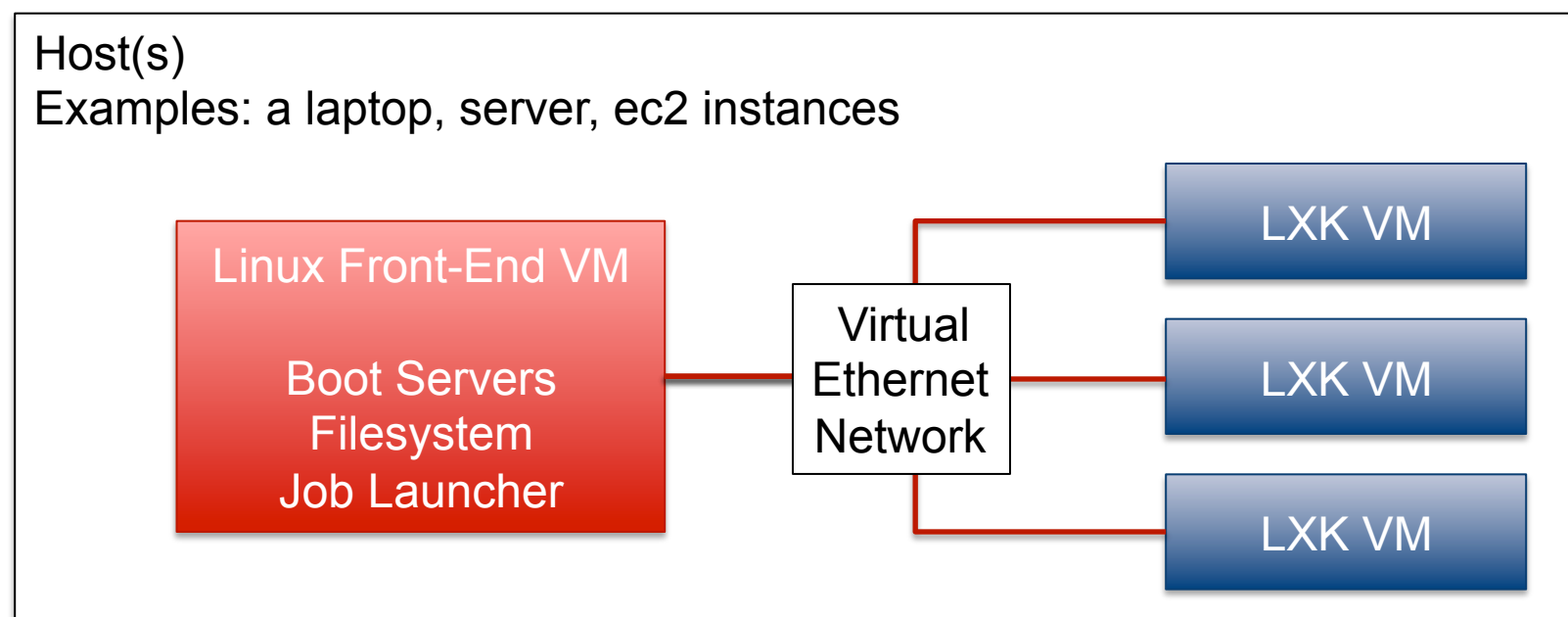
# Porting / Re-Architecting RCR for LXX



- Current RCR architecture (pre-XPRESS)
  - User-level daemon with root privilege
  - Makes high-frequency system calls to Linux (polling)
  - Relatively high overhead (~20% of a Xeon core)
- Goal: Dramatically reduce overhead, eliminated daemon
- Plan: Split RCR into two pieces
  - 1) LXX internal module, timer interrupt used to poll counters, include simple logic to calculate derived metrics / statistics
  - 2) User-level library, implements interfaces for configuring LXX module, reading collected data via shared memory mapping with kernel, interface with RIOS/APEX
- Targeting end of April for initial implementation

# Virtual L XK Development Environment

- Needed easier way to deploy L XK to collaborators
  - Use bundled set of virtual machines: 1 Linux, N L XK
  - Goal to provide functional development environment, reasonable performance
  - Initial development in FY13, continuing in FY14
    - Brian Kocoloski (U. Pittsburgh) – Portals4 networking and job launch
    - Jorge Cabrera (Florida Intl. U.) – I/O forwarding layer from Kitten to Linux



# Community / Vendor Interactions

- Met with Intel March 11, 2014 (telecon)
  - OS Pathfinding group evaluating Kitten for mOS
- Met with AMD February 5, 2014 (SNL/NM)
  - Update/discussion on Hobbes and Kitten
- Met with ARM December 18 (Austin)
  - Update/discussion on Kitten ARM port
- Hosted Hermann Härtig at SNL/NM November 14-15
  - Learned about European Exascale OS R&D
- Kitten/LXK overview invited talk at Hermann Hartig's HPC OS workshop in Israel, Dec . 11, 2013 (via Skype)
- Kitten/LXK overview talk to Argo team, Oct. 28, 2013

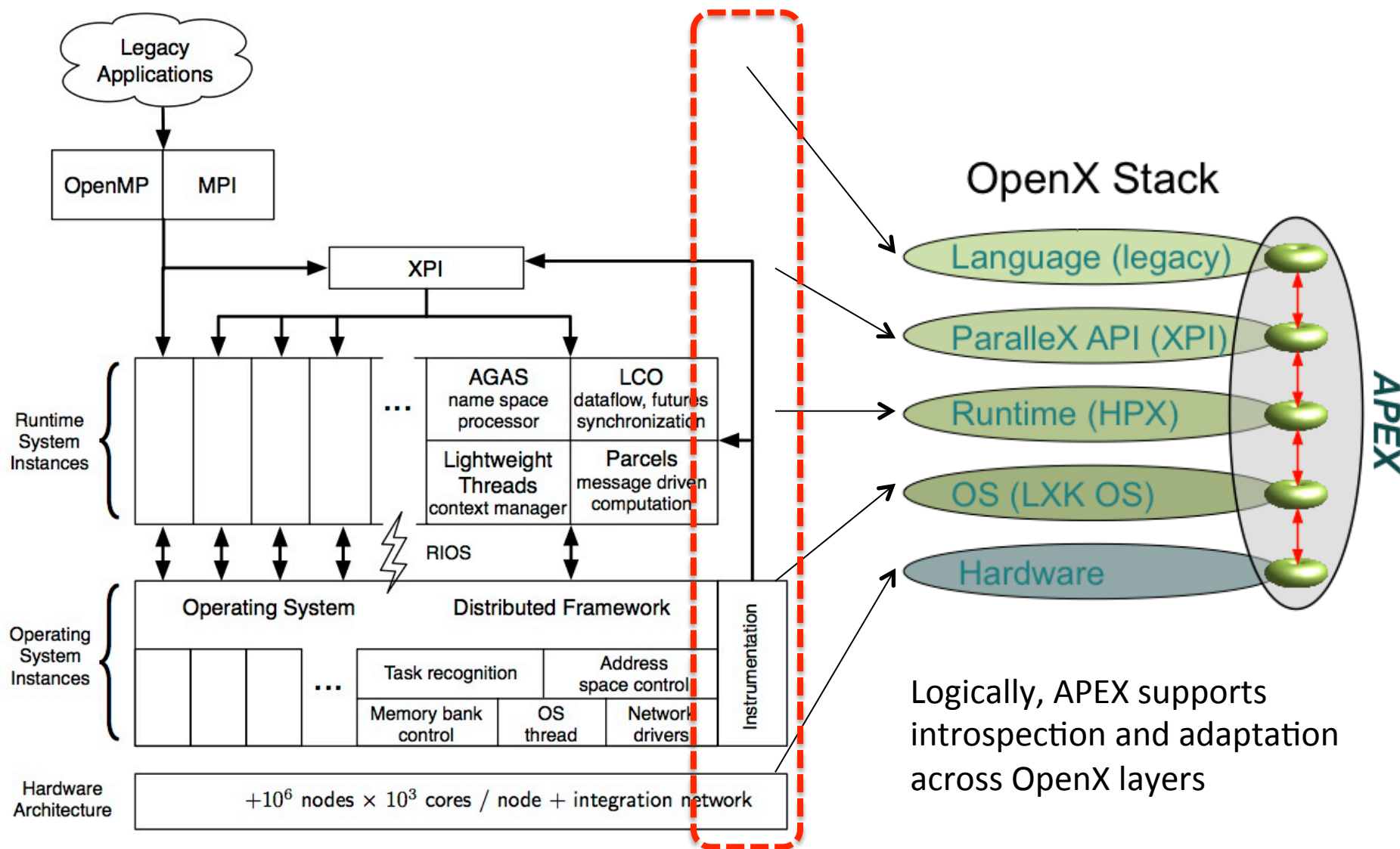
# Conclusion

- XPRESS aiming to increase synergy of compute node OS kernel and user-level runtime systems
  - Today: Runtime must work around host OS, assume worst case
  - Vision: Runtime cooperates with host OS, delegated more control
- Developing RIOS specification (Runtime Interface to OS)
  - Provides runtime with guarantees about resource ownership and behavior
  - Two-way interfaces
    - Runtime tells OS what it needs, OS tells runtime what it gets
    - OS can ask runtime to give up resources, gives runtime time to react
  - Event-based notification of dynamic events (e.g., P-state change)
- Ongoing L XK software development in context of overall XPRESS project

# XPRESS : Introspection overview

Allen D. Malony, Allan Porterfield,  
Sameer Shende, Kevin Huck, Nick  
Chaimov

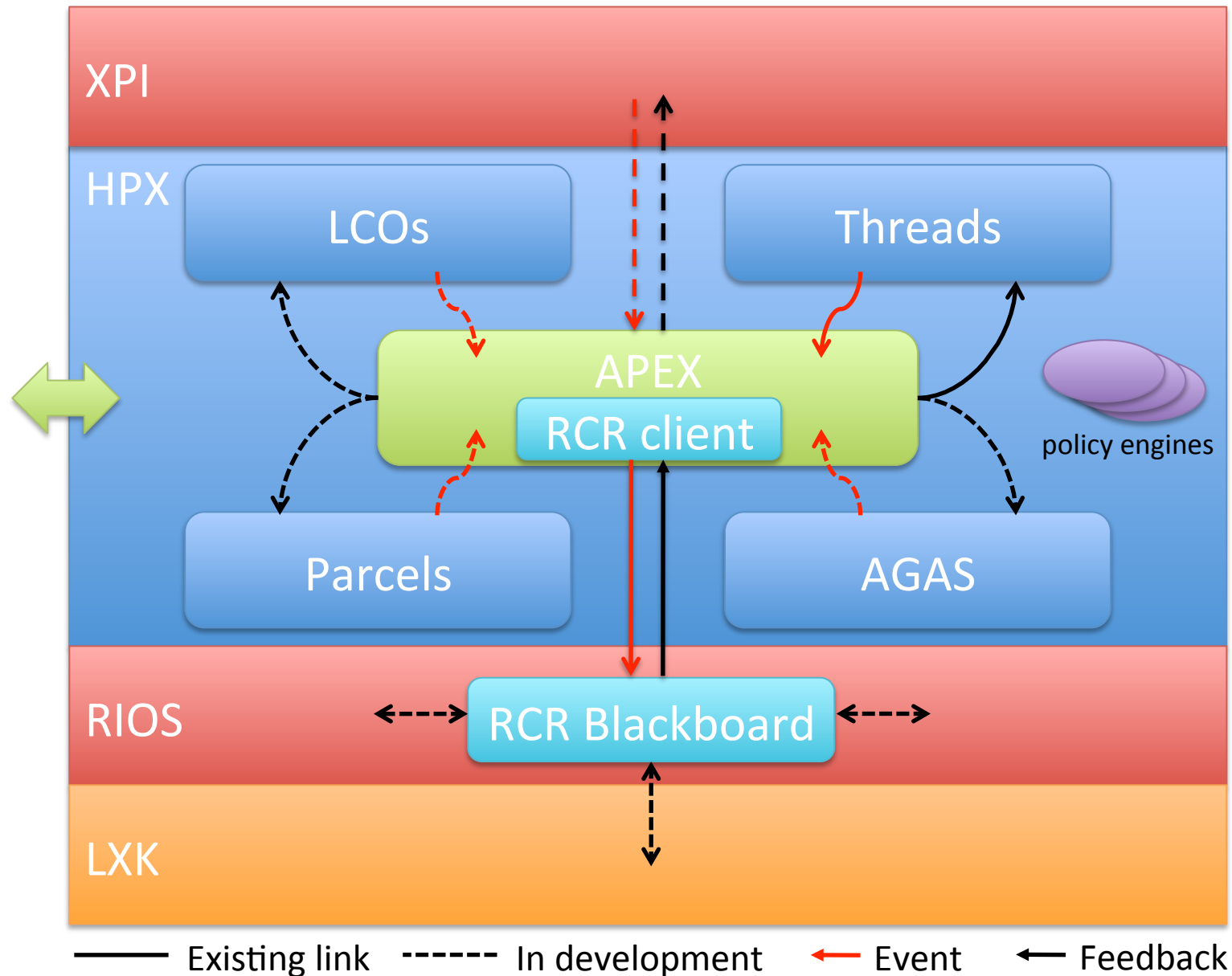
# OpenX Architecture and APEX/RCR's Role



# Introspection Overview

- The XPRESS project provides the opportunity for all layers of the software stack to not just inter-operate, but also enable introspection between layers
- The APEX component encapsulates the mechanisms for introspection and for dynamic adaptation at runtime
- APEX is designed to maintain and/or observe the running performance state of all XSTACK layers:
  - Application level (e.g., XPI interface, events in HPX)
  - System level (e.g., RIOS interface, LXK, hardware)
- APEX provides an interface for:
  - Submitting state changes (observations to analyze)
    - Observable HPX state changes
    - RCRblackboard to observe OS/system changes
  - Registering adaptive policy criteria with a *Policy Engine*
  - Produce events to change performance
    - Events to HPX
    - Use RIOS to inform OS

# XPRESS Interfaces : APEX interactions



# APEX architecture

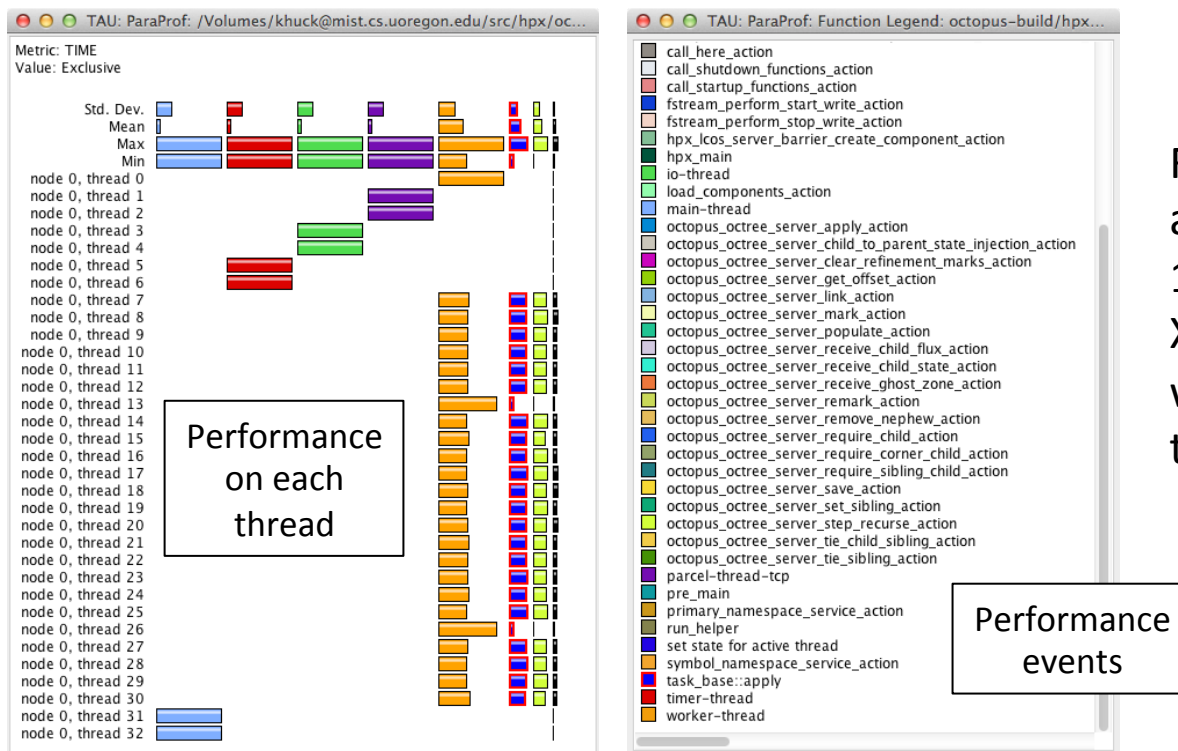
- Two key components provide APEX infrastructure
- Event-driven execution model
  - Thread Manager currently supported
    - thread registration, counter sampling, activity start/stop events
  - Plan to add LCOs, Parcels, AGAS, XPI events
- Periodic interruption / interrogation of the overall system state by an out-of-band thread
  - Process-wide view of thread activity, system health
- Global APEX operation provided HPX
  - Enables APEX to develop monitoring, in situ analytics, and policy responses that will run across system

# HPX ITT support

- HPX3 Threads are instrumented with Intel<sup>®</sup> Instrumentation and Tracing Technology (ITT)
- APEX implemented the ITT API to capture events in the HPX3 Thread Scheduler
  - Developed ITT support in TAU
- Provides “hooks” into HPX components
- For other components, to be replaced with simpler APEX event API

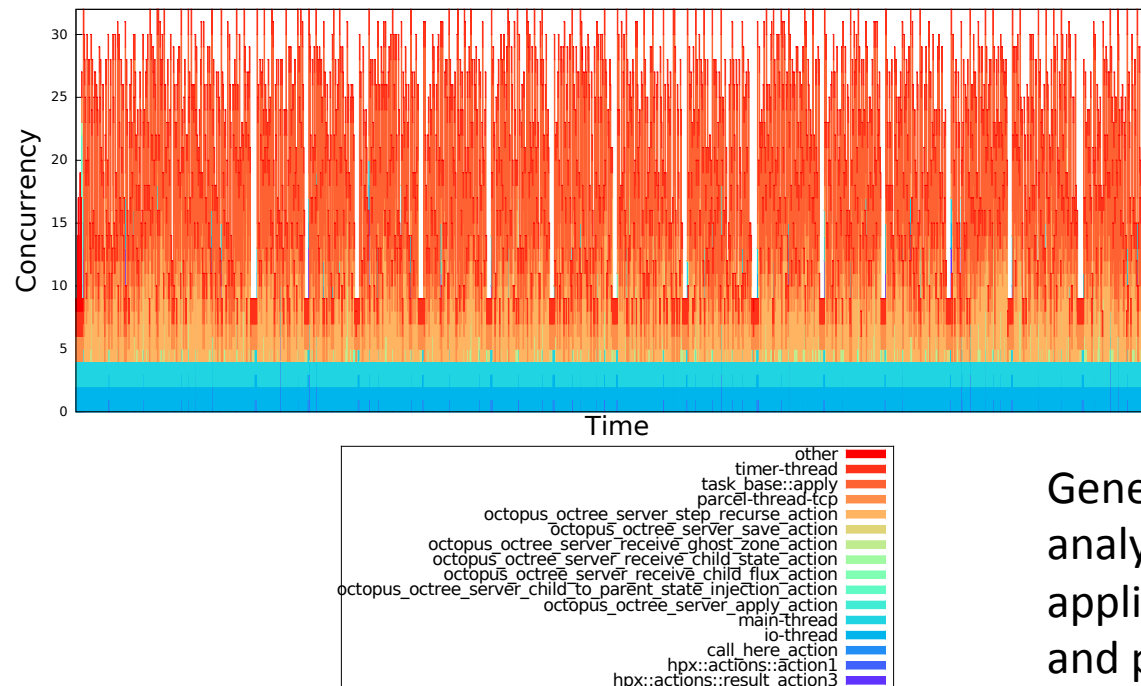
# TAU Integration

- The TAU Performance System has been integrated into APEX as an event listener, providing both profile and trace data collection for post-mortem performance analysis



# Concurrency Example

- Test APEX event handler and periodic interrogation support
- APEX *start* and *stop* events are passed from HPX Thread Scheduler
  - Plugin responds to events
  - Maintains a timer stack for each thread
- Plugin also periodically interrogates state of all threads
- Results in timeline of concurrency activity



General objective is to capture and analyze the dynamic state of application to evaluate behavior and possible take actions to improve

# RCR Integration

- Provides “whole node” view of performance state
- RCR has the same lifetime as the Operating System, most of APEX is associated with an application’s lifetime
- RCRblackboard provides shared memory region for communicating across process boundaries
- LXX monitors OS resources, hardware resources, power consumption, and so on
  - Reads, writes to/from RCRblackboard
- The APEX RCR Client runs in user space, monitors OS & HW state, reports HPX, XPI and Application state (through API)
- Policy Engine determines when to respond to state changes
  - As specified in policy definitions

# RCR

- Integrated with LXX
  - 2 components
    - LXX module – writes system counter values and derived metrics into shared memory page (RCRblackboard)
    - RIOS library routines – make RCRblackboard values visible to user level applications
  - Counters/metrics tracked identified during LXX module initiation
  - RIOS provides mechanism to pass information(events) back to the OS through RCRblackboard – allowing introspection to dynamically modify OS behavior

# Policy Engine

- The Policy Engine subcomponent will also be one of the internal event listeners registered in APEX
- Policy is set in APEX by registering test and action functions with the engine
- Either periodically or on an event, the Policy Engine executes the test functions, and if they return true the action function is executed
- Any XSTACK layer component will have the capacity to respond to the performance status of any other layer as necessary

# XPI, Application Interface Support

- XPI will also report state changes to APEX, respond to policy actions when appropriate
- Application developers may opt to define policy
- Need to specify user-level API for defining, registering application policy

# Future Plans

- Refine APEX API
- Add event reporting from LCOs, Parcels, AGAS
- Add event reporting from XPI implementation
- Define API for defining policy by application
- Redesign RCR Client interface in APEX
- **Identify, specify default policy for OpenX**
  - Non-trivial, requires close interaction with XPRESS partners to understand stack interdependencies
  - Hardware dependent
- Expand RIOS API to support communication with RCRblackboard
- Prototype L XK module to fill RCRblackboard system-level data

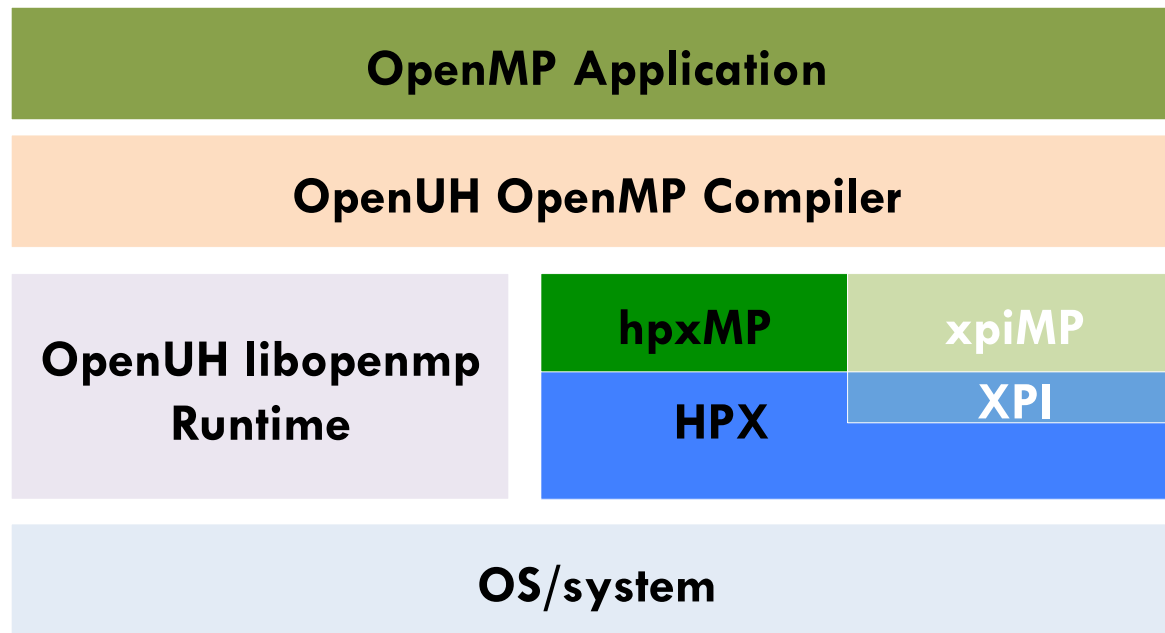
# Legacy migration XPRESS review

Barbara M. Chapman, Edgar Gabriel,  
Yonghong Yan, Jeremy Kemp, and Priyanka  
Ghosh

University of Houston

April 2014

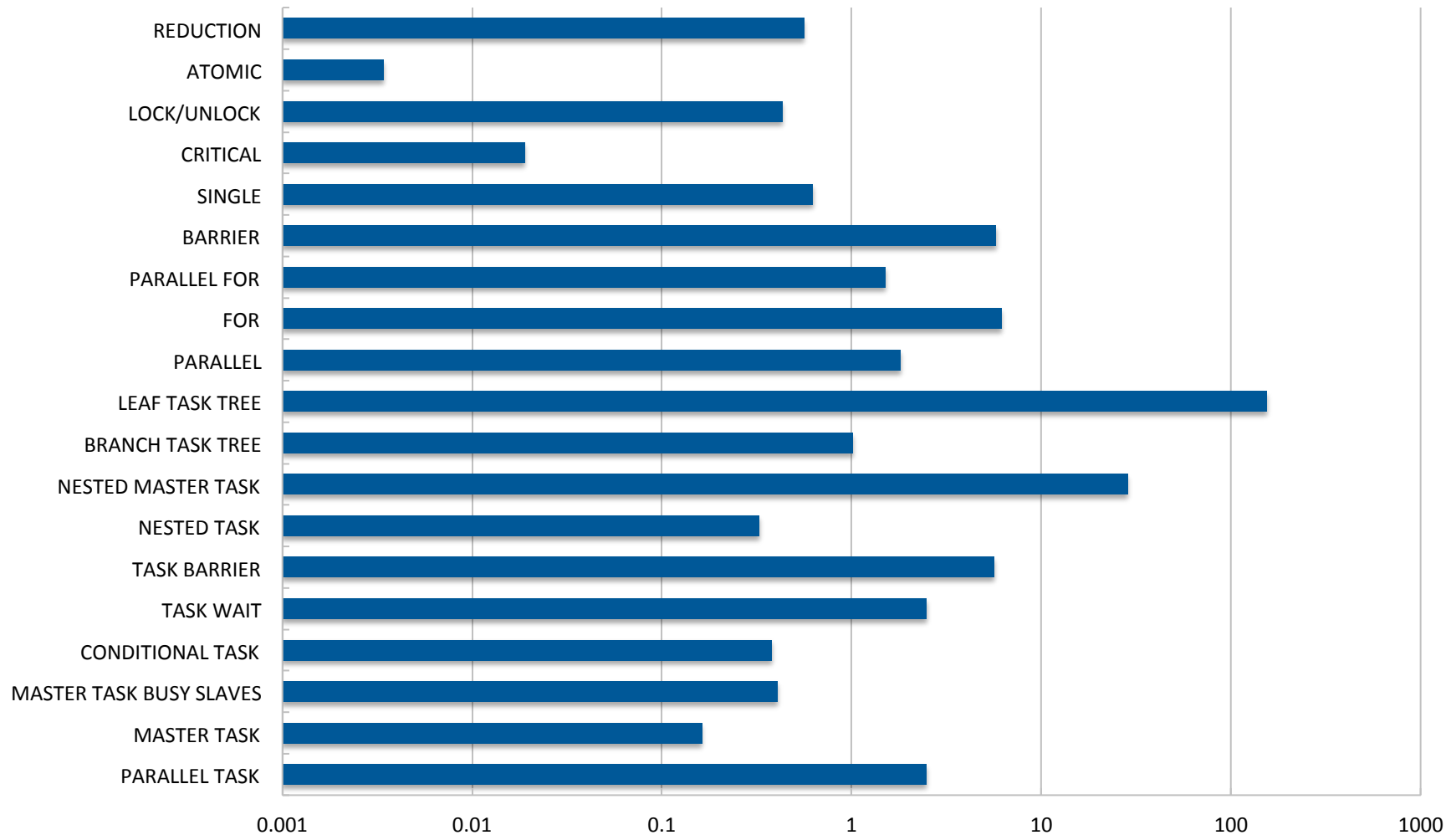
# Legacy migration: replacing OpenMP runtime with HPX and XPI



- Next steps
  - OpenMP runtime on top of XPI
  - OpenMP interoperability with HPX and other runtime

# Microbenchmarks: hpxMP vs OpenUH libopenmp

EPCC Syncbench and Taskbench: <1 means hpxMP is better than libopenmp



# OpenMP for accelerators

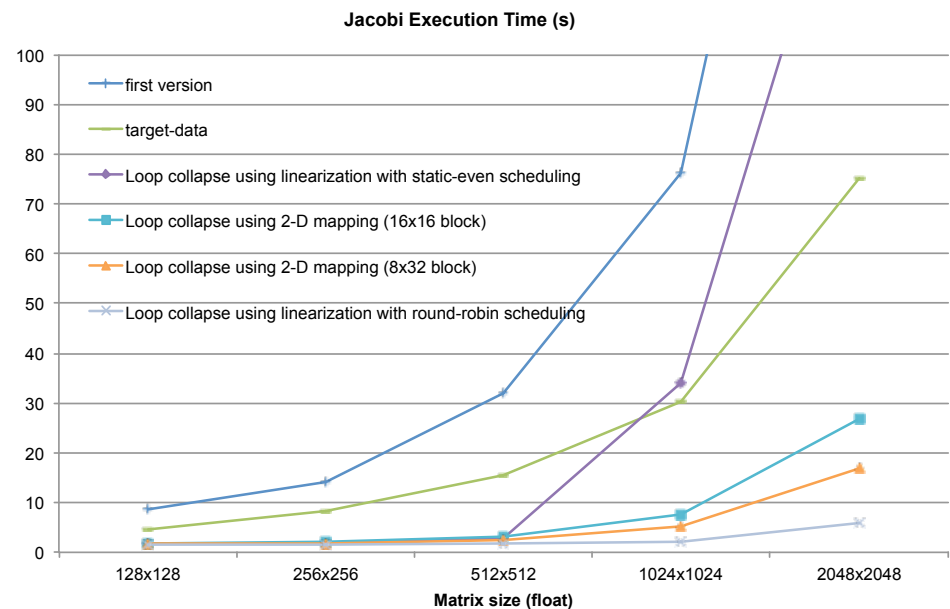
```
#pragma omp target data device (gpu0) map(to:n, m, omega, ax, ay, b, \
    f[0:n][0:m]) map(tofrom:u[0:n][0:m]) map(alloc:uold[0:n][0:m])
```

```
while ((k<=mits)&&(error>tol))
{
    // a loop copying u[][] to uold[][] is omitted here
    ...
    #pragma omp target device(gpu0)
```

```
#pragma omp parallel for private(resid,j,i) reduction(+:error)
```

```
for (i=1;i<(n-1);i++)
    for (j=1;j<(m-1);j++)
    {
        resid = (ax*(uold[i-1][j] + uold[i+1][j])\
            + ay*(uold[i][j-1] + uold[i][j+1]) + b * uold[i][j]
        u[i][j] = uold[i][j] - omega * resid;
        error = error + resid*resid ;
    } // rest of the code omitted ...
}
```

Early Experiences With The OpenMP Accelerator Model; Ch and Barbara Chapman; International Workshop on OpenMI



# Compiler support for OpenMP and OpenACC

- Need to achieve coalesced memory access on GPUs

```
#pragma acc loop gang(2) vector(2)
for ( i = x1; i < X1; i++ ) {
  #pragma acc loop gang(3) vector(4)
  for ( j = y1; j < Y1; j++ ) {..... }
}
```

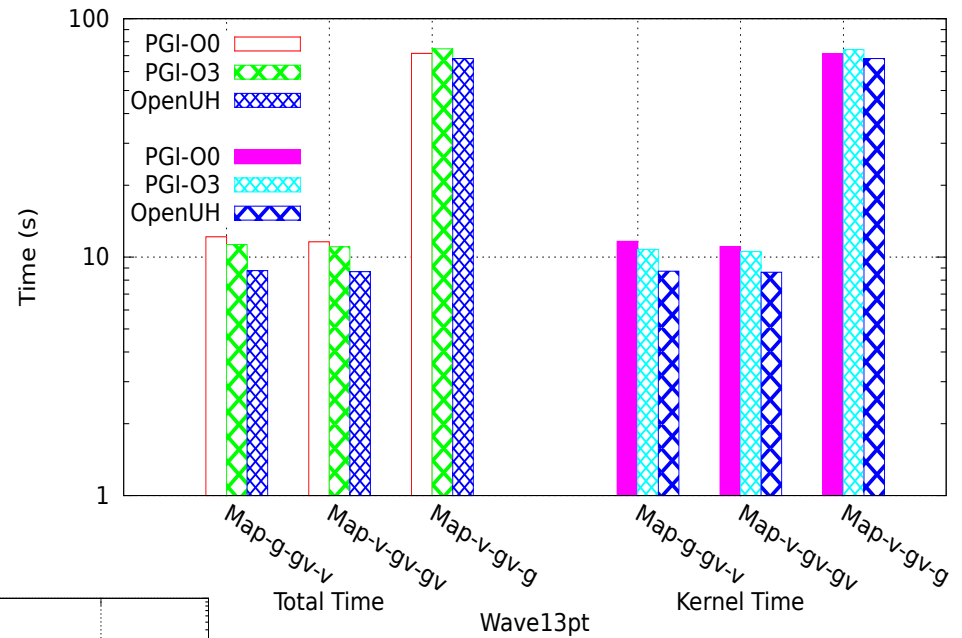
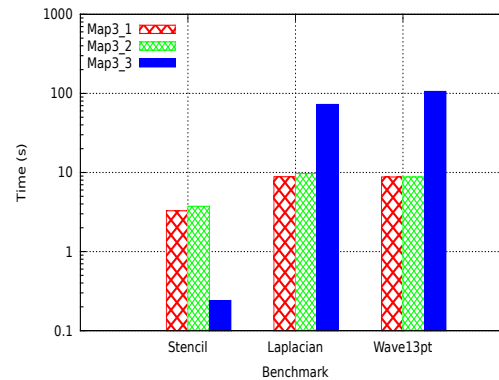
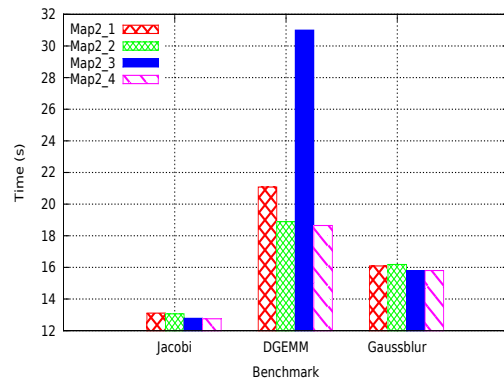
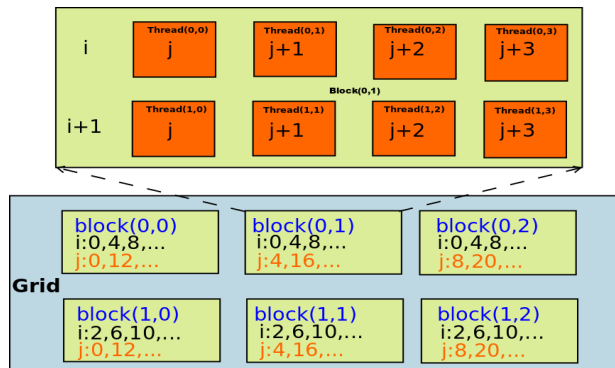
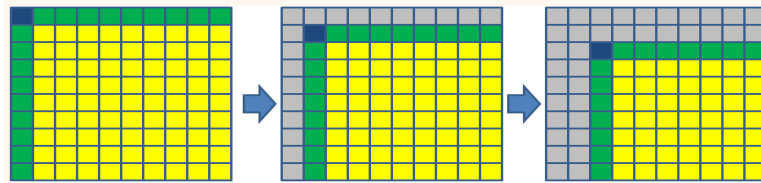
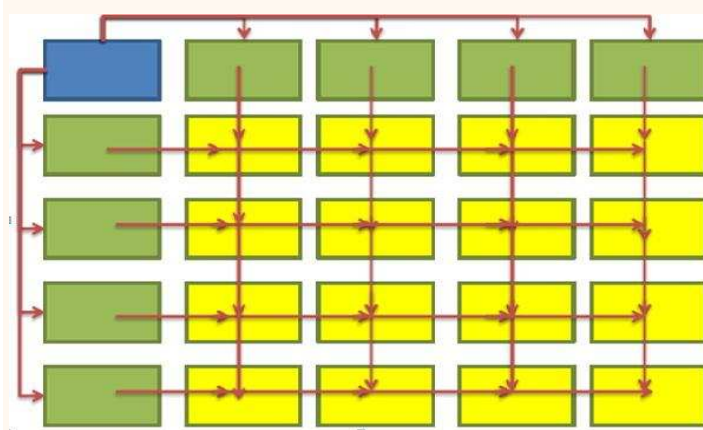


Figure: Wave13pt

Fig. 9: Double nested loop mapping.

Fig. 10: Triple nested loop mapping.

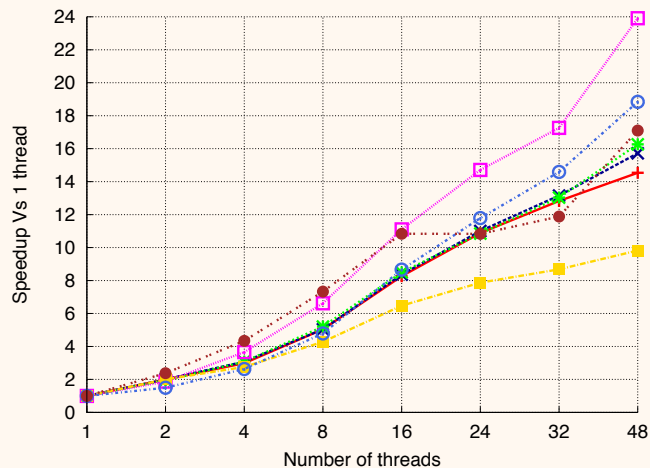
# Data-driven async tasks



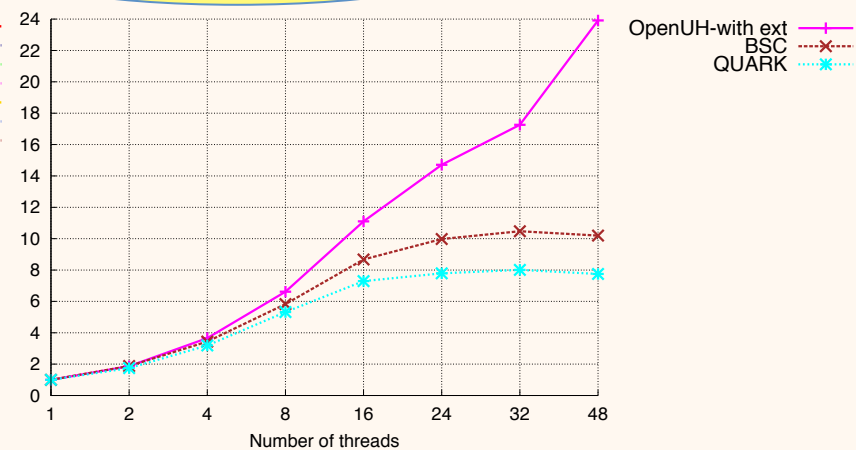
```

1  #pragma omp parallel
2  {
3    #pragma omp master
4    {
5      for ( i=0; i<matrix_size; i++ ) {
6
7        /**** Processing Diagonal block *****/
8        ProcessDiagonalBlock (.....);
9
10     for ( i=1; i<M; i++){
11
12       #pragma omp task out(2*i) /*** Processing block on column **/
13       ProcessBlockOnColumn (.....);
14
15       #pragma omp task out(2*i+1) /*** Processing block on row **/
16       ProcessBlockOnRow (.....);
17     }
18
19     /*** Elimination of Global Synchronization point *****/
20
21     /**** Processing remaining inner block *****/
22     for ( i=1; i<M; i++){
23       for ( j=1; j<M; j++){
24         #pragma omp task in(2*i) in(2*j+1)
25         ProcessInnerBlock (.....);

```



GNU  
 Intel  
 OpenUH-without ext  
 OpenUH-with ext  
 SUN-Oracle  
 PGI  
 OmpSs



OpenUH-with ext  
 BSC  
 QUARK

# Support for Open MPI on HPX/XPI

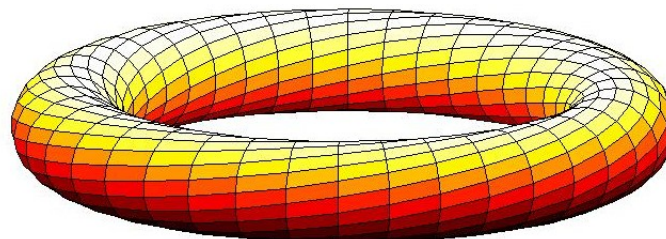
- Work focuses on taking advantage of HPX features in Open MPI Runtime
  - Application would be able to utilize features of MPI and HPX simultaneously
- Two options being evaluated:
  - Replace entire RTE of Open MPI with a slim layer based on HPX -> daemonless approach
  - Add new components in ORTE to support HPX
- Operational prototype expected by the end of the summer

# XPRESS Apps Efforts

- Scope:
  - Explore app development alternative to “traditional MPI+X”.
  - Question: Can a qualitatively different approach (Parallex-based):
    - Exploit untapped parallelism?
    - Improve expressability?
    - Improve productivity?
    - Get us to Exascale and beyond?
- Efforts:
  - Miniapp focus (far upstream from when real apps are available).
  - Broad sampling of app domains & algorithms:
    - Plasma physics & particle-in-cell (PIC)
    - Nuclear engineering & finite volume/eigensolvers.
    - Shock physics & finite element/explicit time integration.
    - Computational mechanics & implicit sparse solvers.

# Recent highlights

- Ramp up of full apps effort complete:
  - Mike Heroux, et. al. – Overall lead, SNL engineering apps.
  - Matt Anderson, et. al. – IU (ongoing).
  - Tom Evans, ORNL, nuclear engineering apps at extreme scale.
  - Alice Koniges, LBL/NERSC, plasma, PIC.
- SPN (Denovo Nuclear Eng miniapp) ready for release.
- HPCG (TOP 500 benchmark code) ported to XPI/HPX.
- MiniAMR: New CTH-like miniapp.
- C/C++ versions of GTC/PIC codes under evaluation.
- Alternative (non-Poisson solve) PIC algorithms.



# Efforts

- Use of Futures:
  - Exploit previously inaccessible, fine-grain dynamic parallelism.
  - Natural framework for expressing data-driven parallelism.
  - SPN, HPCG, MiniAMR.
- Better than MPI:
  - Beyond functional mimic of MPI.
  - AGAS: Truly adaptive mesh refinement.
- Overarching goal: Demonstrate that Parallelex-based approaches:
  - Work.
  - Superior to MPI+X in one or more metric:
    - Performance: Extracting latent parallelism.
    - Portability: Performance obtained from system's underlying runtime.
    - Productivity: Easier to write, understand, maintain.

<http://xstack.sandia.gov/xpress>